3ETI, Examen [CSC2]
Developpement Logiciel en C
CPE Lyon

2013-2014 (2ème session)
durée 2h

Tous documents et calculatrices autorisés.

Répondez aux questions sur une copie séparée

Le sujet comporte 4 pages

Le temps approximatif ainsi que le barème sont indiqués pour les grandes parties. Notez que le barème est donné à titre purement indicatif et pourra être adapté par la suite.

En cas de doute sur la compréhension de l'énoncé, explicitez ce que vous comprenez et poursuivez l'exercice dans cette logique.

Note préliminaire:

- Toutes les questions concernent le language de programmation C dans le cadre du developpement de logiciels sur architecture PC standard.
- Nous vous invitons à commenter le code et les réponses. En particulier, en cas d'ambiguité de compréhension d'une question, ajoutez toutes remarques et illustrations supplémentaires permettant d'expliquer votre démarche.
- Sauf mention contraire explicite, on supposera que l'on dispose d'un système Linux standard fonctionnant correctement sur un PC récent 32 ou 64 bits (identiques aux conditions de TP des PC de CPE: *int* étant encodé sur 4 octets, pointeurs étant encodés respectivement sur 4(/8) octets sur 32(/64) bits).
- On supposera que le code C est compilé avec une version récente de gcc sous la norme C99 (ou ultérieure) identique aux conditions de TP des PC de CPE.
- On supposera dans chaque cas que les `#include` des en-tête standards nécessaires à la bonne compilation et exécution des programmes décrits sont correctement installés et appelés (ex. stdio, stdlib, string, math, etc).

1 Organisation mémoire

[45min max] 7 points

On suppose que l'on exécute le code suivant sur un système classique 64 bits little endian (Un `int` est codé sur 4 octets, un `short` sur 2 octets, un pointeur sur 8 octets). On rappelle que `printf(``%x'',...);` affiche la valeur d'une variable en hexadécimal.

```
struct info_maison
{
  unsigned short hauteur;
  unsigned short largeur;
};

void complete_info_maison(struct info_maison* m,int h,int l)
{
  assert(m!=NULL);
  m->hauteur=h;
  m->largeur=l;
}

struct village
{
  struct info_maison maison[3];
  char nom[28];
  void* image;
  unsigned int nbr_habitants;
};

void* load_image(const char* filename)
{
  //chargement de l'image
  //...
  // On suppose que l'adresse retournee vaut 0x12457865
  return (void*)0x12457865;
}

int main()
{
  struct village v;
  complete_info_maison(&v.maison[0],12,6);
  complete_info_maison(&v.maison[1],14,8);
  complete_info_maison(&v.maison[2],8,16);
  strcpy(v.nom,"Dompierre-sur-Mer");
  v.image=load_image("image.png");
  v.nbr_habitants=15;

  void *temp=&v;
  short *u=temp;
  printf("%x\n",u[12]+u[20]);

  return 0;
};
```

On compile ce programme et on l'exécute.

Question 1 *Qu'affiche l'exécution de ce programme sur la ligne de commande ?*

2 Préprocesseur

[20 min max] 4 points

Soit le fichier d'en-tête dénommé `fichier.h` suivant:

```
#ifdef ENTREE
struct etudiant
{
  VAR1;
  VAR2;
};
#endif

#ifndef ENTREE
#define ENTREE
#ifdef D1
#define VAR1 char nom[20]
#define VAR2 float note
int g=12;
#else
#define VAR1 char nom[15]
#define VAR2 int note
int g=17;
#endif
#endif
```

On suppose que l'on dispose également du fichier suivant (`main.c`) situé dans le même répertoire que `fichier.h`.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

???

int main()
{
  struct etudiant e;
  e.note=g;
  printf("%lu\n",sizeof(e.nom));
  printf("%f\n",e.note);
  return 0;
}
```

Question 2 *Par quelle(s) ligne(s) de directive(s) préprocesseur doit-on remplacer les* `???` *pour que ce programme puisse compiler sans Warnings (options Wall et Wextra activées) ?*

3 Méthodologie de programmation

[55min max] 9 points

On souhaite modéliser un ensemble de voitures et l'évolution de leurs consommations en essence. On considère les choix suivants.

- Une voiture est modélisée par une marque, son nombre de kilomètres au compteur, et l'essence restante dans son réservoir.
- Il y a 4 marques différentes: *Volvo, Peugeot, Citroen, Volkswagen*.
- La consommation des marques (en L pour 100km) sont les suivantes
  - Volvo: 4.2L
  - Peugeot: 3.5L
  - Citroen: 4.4L
  - Volkswagen: 6.5L

L'appel à une fonction `voiture_roule` permet de modéliser le fait qu'une voiture roule sur un certain nombre de kilomètres (donné en paramètre de cette fonction). La voiture met à jour son nombre de kilomètres et la quantité d'essence. Si la quantité d'essence le permet, la voiture roule l'ensemble des kilomètres demandés. Sinon, la voiture avance sur autant de kilomètres que possible (jusqu'à n'avoir plus d'essence), puis un message indique en ligne de commande qu'une voiture est en panne. Le message doit indiquer la marque de la voiture.

On suppose que le code suivant permetterait de modéliser 3 concurrents qui roulent 20 étapes de 50 kilomètres. Dans le cas présent, la voiture Volvo fini par être en panne d'essence sur la dernière étape.

```
int main()
{
  struct voiture concurrents[3];
  voiture_initialise(&concurrents[0],Volvo,50000,45);
  voiture_initialise(&concurrents[1],Peugeot,150000,55);
  voiture_initialise(&concurrents[2],Volkswagen,2000,75);

  int k=0;
  for(k=0;k<20;++k)
    {
      int k_concurrent=0;
      for(k_concurrent=0;k_concurrent<3;++k_concurrent)
          voiture_roule(&concurrents[k_concurrent],50);
    }

  return 0;
}
```

Question 3 *Ecrivez le code C que vous mettriez en place pour modéliser ce fonctionnement. Votre code devra être compatible avec la structure de la fonction* `main()` *fournie. Votre code devra respecter les critères de bonnes programmation.*